student.component.html

```
<p> student works ! </p>
<h1> Student Component </h1>
 <div>
    <ol>
      <li> Virat </li>
      <li> Sachin </li>
      <li> Dhoni </li>
      <li> Rohit </li>
      <li> Gautham </li>
      <li> Yuvi </li>
    </ol>
 </div>
```

Output:-

Routing Example

student | student-details

student works !

Student Component

1. Virat
2. Sachin
3. Dhoni
4. Rohit
5. Gautham
6. Yuvi

student-details-component.html

```
<p> Student - Details </p>

<div>
  <p> Virat </p>
  <p> Cricket </p>
  <p> Batsman </p>
</div>
<div>
  <p> Sachin <br> Cricket <br>
      AllRounder </p>
</div>
<div>
  <p> Dhoni <br> Cricket <br>
      Batsman, Keeper </p>
</div>
```

Output:-

Routing Example

student | student-details

student-details works !

Virat
Cricket
Batsman

Sachin
Cricketer
All Rounder

Dhoni
Cricketer
Batsman, Keeper

student-details.component.css

```
div{
 border : solid;
 margin-bottom : 10px;
}
```

* localhost: 4200 displays the root component (app-component)
* In the URL localhost: 4200/student will display the student component data in the browser.
* Localhost: 4200/student-details will display the student detail component on the browser
* Whenever we write localhost: 4200 in the URL the second half page on the browser will be empty.

Angular Routing

* An angular router is an offical routing library, written and maintained by the Angular Core Team.
* It activates all the required Angular Component to compare a page when a user navigates to certain URL
* It lets users navigate from one page to another page without page reload.
* Whenever the users clicks on link it will go into routes array and it will search the path
* Angular provides easy way to create and work with components in its SPA.

Router Outlet

* Router Outlet is dynamic component that the router uses to displays views based on router navigation.
* Router outlet is a routing component

```
<router-outlet></router-outlet>
```

* The router-outlet is the selector for the RouterLink directive that turns user click into the user navigation
* You can assign a string to the routerLink
* This directive generates our link based on the route path.

Redirecting Routes

* A redirect route that translates the initial relative URL to the desired default path.
* When application starts, it navigates to the empty route to by default. We can ~~chang~~ configure the router to redirect to a named route by default.
* A routes has no routes until you configure it.
* Export const routes : Routes = [
  { path: '', redirectTo: 'component-one' (student), pathMatch: 'full' },
  { path: 'student', component: StudentComponent },
  { path: 'employee', component: EmployeeComponent },];

Rules

* Empty path in the first route represents the default path for the application.
* A redirect route requires a pathMatch property to tell the router how to match a URL
* We all need pathMatch : 'full' property so Angular knows it should be matching exactly the empty string.

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { studentComponent } from './student/student.component';
import { studentDetailsComponent } from './studentdetail/studentDetails
                                          Component';
const routes: Routes = [
{ path: '', redirectTo: 'student', pathMatch: 'full' },
{ path: 'student', component: studentComponent },
{ path: 'student-details', component: studentDetailsComponent },
];
@NgModule({
imports: [RouterModule.forRoot(routes)],
exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.html

```
<h1> Routing Example </h1>
<div>
  <a [routerLink]="['/student']">student</a>
  <a [routerLink]="['/student-details']">student details</a>
</div>
```

Output

localhost: 4200/student

Routing Example

student | student-details

student works!

student Component

1. Virat
2. Sachin
3. Dhoni

Output

localhost: 4200

Routing Example

student student-details

student works!

Student Component

1. Virat.
2. Sachin.
3. Dhoni.

Wildcard Route:-

* Wildcard route to intercept invalid URL's and handle them gracefully
* A wildcard route has a path consisting of two asterisks {**}
* It matches every URL, the route will selects this route it can't match a route earlier in the configuration. A wildcard route can navigate an existing route to a custom "404 not found" component or redirected to an existing route.

```
{
   path: '**', component: PageNotFoundComponent
}
```

* If the entire router configration is processed and there is no match, router navigation will reach default.

Rules:-

* If you add a wildcard route as the first route, no other would be reached and the wildcard route would always be matched
* As a result, you should always add a wildcard route as the last route in your router configuration
* Create a pagenot found Component using terminal

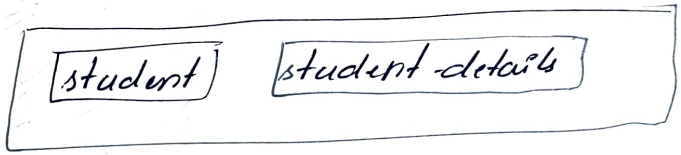
app-routing.module.ts

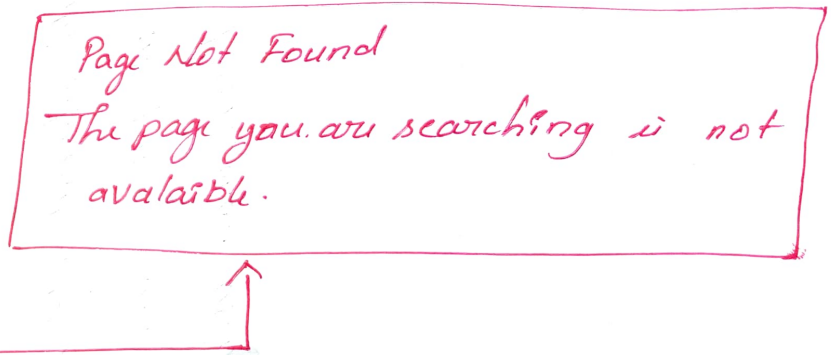
```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { studentComponent } from './student/student.Component';
import { studntDetailsComponent } from './student-details/studnt-details.
                                                    Componnt';
import { PageNotFoundComponent } from './pageNotFound/pageNotFound
                                                    Component';

const routes: Route=[
{ path: 'student', component: studentComponent },
{ path: '', redirectTo: 'student', matchMatch: 'full' },
{ path: 'studntDetails', component: studentDetailsComponent },
{ path: '**', component: PageNotFoundComponent }
];
@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})
export class AppRoutingModule { }
```

Output:-

localhost:4200/login <== not present in the component

Routing Example





Note: If the page not found component is placed at the beginning then it will always show the error on the browser. So we have to place it at the end.